**CSCE 313 509**

**Programming Assignment #0**

**Report**

Vidhur Potluri

UIN: 626007235

**Setup**

To set up the development environment, I installed Virtual Box and Ubuntu. I had to enable virtualization in the BIOS settings of my computer. Once I had the VM ready, I installed g++, GDB and clion using the standard commands of the Linux terminal.

**Basic and common steps**

After I downloaded buggy.cpp, I compiled the program and found errors that were caused by the wrong syntax being used and not including certain libraries. I added the vector library to the file and the std namespace to the program in blank A to fix this. I changed blank B to ‘public’ to allow access to the class outside of its member functions. I also corrected the syntax to be used to access the data members of the pointers. Each time I compiled the program, I added -g to load the symbol table. After I finished the debugging with both tools, I added a for loop to delete the nodes to prevent memory leaks.

**Debugging**

GDB: After I ran the executable buggy file with GDB, it displayed a segmentation error at line 16 (originally line 15) in the create\_LL function. The gdb backtrace tool also specifies that the function was called in the main function. I then set a breakpoint at line 16 using the instruction ‘break buggy.cpp:16’ and accessed the value in myList[i] using ‘print myList[i]’. The tool couldn’t access the value since the node wasn’t initialized. I fixed this by filling in blank C with ‘mylist[i] = new node’. After I fixed this error and compiled the program again, GDB detected an error at line 29 (originally 28) inside sum\_LL. The function tried to access a node at a position larger than the size of the vector. The second part of the function (linked list) assigns the ‘next’ node data member of a node. However, the node at position i – 1 of mylist should not have a ‘next’ data member. I fixed this by changing the for loop condition to ‘i+1 < node\_num’.

AddressSanitizer: The first error that AddressSanitizer finds is located at line 16 (originally line 15). The tool also specifies that the error is in the function create\_LL. The tool also provides a hint as to what the address of the variable points to. The tool stops the program execution after the first runtime error is detected. The second error is then detected at line 22 (originally line 21). It detects heap-buffer-overflow in create\_LL function. The tool also displays an address table.

**Corrected Code**

#include <iostream>

#include <vector>

using namespace std;

class node {

public:

int val;

node\* next;

};

void create\_LL(vector<node\*>& mylist, int node\_num){

mylist.assign(node\_num, NULL);

//create a set of nodes

for (int i = 0; i < node\_num; i++) {

mylist[i] = new node;

mylist[i]->val = i;

mylist[i]->next = NULL;

}

//create a linked list

for (int i = 0; i + 1 < node\_num; i++) {

mylist[i]->next = mylist[i+1];

}

}

int sum\_LL(node\* ptr) {

int ret = 0;

while(ptr) {

ret += ptr->val;

ptr = ptr->next;

}

return ret;

}

int main(int argc, char \*\* argv){

const int NODE\_NUM = 3;

vector<node\*> mylist;

create\_LL(mylist, NODE\_NUM);

int ret = sum\_LL(mylist[0]);

cout << "The sum of nodes in LL is " << ret << endl;

//Step4: delete nodes

for (int i = 0; i < NODE\_NUM; i++) {

delete mylist[i];

}

}